JC542 U.S. PTO 09/209162 12/10/98

KidCode® Application Programming Interface (API)

This API defines the data and function calls that are used for communication between the KidCode Main Email program and installable components. Each installable component can be one of two types:

- mailbox browser/editor component
- message authoring/display component

KidCode Main Email application may communicate with another mail server such as an SMTP compliant server to retrieve and store email messages. Alternatively, the Email Main program may include code for many of the functions normally associated with a mail server program. Whether in conjunction with a mail server, or on its own, the Email Main program handles all functions associated with sending and receiving email messages. This includes reading and writing mailbox files to/from permanent storage or other mail servers on a network (e.g. using POP3), finding and verifying network addresses, and sending mail messages to other servers on a network.

The Main Email Program also provides a GUI that provides interaction with a user for those functions that are directly associated with storage and transfer of electronic mail messages and mailboxes. In particular, the Main Email program includes buttons and/or menu items that allow a user to:

- **Send** (a message),
- **Reply** (to a message),
- **Open** (a message or a mailbox),
- **Delete/Trash** (messages or mailboxes),
- **Save** (a message to an alternative mailbox)
- **Print** (a message)

The Main Email Program also handles all data bundling and unbundling that may be required to transform the message data used by a message authoring component into a fully MIME compliant message type. This way each message authoring component can handle data in a format most convenient to it and all MIME parsing and details associated with protocol compliance can be centralized in the Main Email application. The only requirement for the message data passed between a message authoring component and the Main Email Program is that the message body data be formatted either as an ASCII string or in a binhex format.

The KidCode Main Email program communicates with installable components in order to execute the commands defined above.

Mailbox browser/editor components
Mailbox components are used to display, edit, and browse mailboxes. Different kinds of users and different types of messaging applications (e.g. fax, traditional email, internet voice) may require very different displays and functionality from a mailbox viewer/editor. Installable mailbox components make it possible to upgrade, select from multiple viewing formats, utilize different mailbox viewer/editors for different users, and in general increase the range of functionality that can be achieved within one basic messaging application program.

Message authoring/display components
Message handler components make it possible to handle an unlimited number of message types. Each message handler component is designed to deal with a specific MIME type of message. The MIME data standard has been designed so that application developers can define new MIME types as needed by labeling these with the "/application-x" prefix. A message handler component can be any program that defines a message MIME type of data that it handles and that implements the callback entry points described in this document. These functions allow the Main Email application to obtain information about the message handler and allows the message handler to respond to standard mail commands such as Send

or Reply, that have been issued by a user through the Main Email interface. Example message handler components included in the KidCode application are an ordinary ascii text message handler, a game called Rebus that allows users to create and respond to graphical rebus messages, an a sample mathematics workbook that allows students and a teacher to send workbook problems to one another.

Global variable naming conventions:

Each movie should name its global variables with a prefix that identifies the movie and a capital "G" for "global". We will keep track of each movie's prefix. For now we have the following identifing prefixes:

component prefix	component	global variable prefix
em_	main movie	emG_
tm_	text movie	tmG_
rm_	rebus movie	rmG_
cm_	connect movie	cmG_
tgm_	text grid movie	tgmG_
pm_	puzzle movie	pmG_
mbx_	mailbox movie	mbxG_

Main Movie Public Data Types

```
em_ComponentType    symbol = #mailbox or #msgHandler

em_UserName  string

em_UserData struct (
                str      UserName
                str      FullName
                str              ReturnAddress
        em_AddressBook           AddressBook
        em_MailboxList           Mailboxes
                str              SMTPHost
                str              POP3Host
                str              Password
)

em_MailboxName  string

em_Mailbox  struct (
        em_mailboxName  boxName
        list of emMailData
)

em_RegisteredUsers  list of em_UserName

em_MailData struct (
        em_Address               To
        em_Address               From
                str              Re
                str              Data
                str              MimeType
                list             MsgBody
)

em_MessageNumber int

em_Mode symbol = #author or #display

em_ComponentInfo struct (
                str              ComponentName
                int              ComponentID
em_ComponentType                 ComponentType
                str              ComponentMIMEType  ; nil if mailbox
)
```

Email Main API Functions

These functions are called by the installable components to access services provided in the KidCode Main Email program.

```
/*****************************************************************
**/
/* emh_getUserMailbox
Return a mailbox data structure for the current user and mailbox name.  This function is
normally called by a mailbox handling component.  Mailbox handling components may use
temporary files to hold mailbox contents but they should never access the users mailbox files.
All access to these files must be obtained through the Main Email program.
*/

em_Mailbox  emh_getUserMailbox (
                em_MailBoxName
)


/*****************************************************************
**/
/* emh_getUserData
Return a data structure with user information. The KidCode Main Email program maintains
all user information and handles user administration functions.  The Main program also
communication with external Mail servers which may contain other user information not part
of the KidCode API.
*/

em_UserData emh_getUserData (
        em_UserName,
)


/*****************************************************************
**/
/* emh_continue
Used by installable components to explicitly pass control back to the Main Email program.
This function is necessary for the Director/Lingo implementation.
*/

void emh_continue  (
        em_ComponentType
)
```

```
/*****************************************************************
**/
/* emh_killComponent
Used by an installable component to inform the Main Email program that it is preparing to
terminate. This allows the Main program to free any memory and/or data structures that have
been allocated to the component.
*/

void emh_killComponent (
)


/*****************************************************************
**/
/* emh_passMessage
Used primarily by mailbox components to pass a message to the Main program so that it can
be displayed by the appropriate message handling component.  Email main takes the message
argument (em_MailData, looks up the Mimetype of the message, and invokes the appropriate
message handler to display the message.
*/

void emh_passMessage  (
        em_MailData,
        em_MessageNumber
)


/*****************************************************************
**/
/* emh_getMessage
Returns the message (em_MailData) with Number MessageNumber from the MailboxName
of the current user.  Can be used by installable components to retrieve specific messages from
the user's mailboxes.

If this is called with the messageNumber set to 0, email main assume the typeOrBoxName
argument is a mimetype and returns a new message structure.  Message handling components
should call emh_getMessage with the number set to 0 and the mimetype whenever a new
message is started.  Normally this should be done whenever an active message is trashed.
*/

em_MailData emh_getMessage (
        em_MessageNumber
        str      typeOrBoxName
)
```

```
/****************************************************************
**/
/* emh_getRegisteredUsers
Returns a list of usernames for the users that are registered with the KidCode system, i.e. that
have been added as users by the User Adminstration part of the Main Email Program. This is
the same list of users that appear in the logon listbox when the program is started up. It may
be used by installable components to create listboxes for filling address fields in messages or
for checking on whether a particular address is external to the system.
*/

em_RegisteredUsers  emh_getRegisteredUsers (
)


/****************************************************************
**/
/* emh_sendMessage
Email Main sends the message argument (em_MailData) by either forwarding to an external
mail server or, if it is a registered KidCode user, writing the message to the user's incoming
mail mailbox.
*/

void emh_sendMessage (
        em_MailData
)



/****************************************************************
**/
/* emh_saveMessage
Email Main saves the message argument (em_MailData) for the currently logged on user by
writing the message to the user's "notes in progress"  mail mailbox.
*/

void emh_saveMessage (
        em_MailData
)
```

```
/***************************************************************
**/
/* emh_disableButton
It is recommended that this function be used carefully. Normally Email Main controls the
state of all the buttons available to users to access message handling of the main program (i.e.
buttons in the purple left hand panel). This function can be used to request that Email Main
disable the button specified by the argument, ButtonName. If the button is disabled - whether
it was already disabled or is disabled as a result of the function call - the function will return
TRUE, otherwise it will return FALSE. The calling component should check on whether the
function call succeeded and proceed accordingly.
*/

em_ReturnValue emh_disableButton (
                str             ButtonName
)



/***************************************************************
**/
/* emh_enableButton
It is recommended that this function be used carefully. Normally Email Main controls the
state of all the buttons available to users to access message handling of the main program (i.e.
buttons in the purple left hand panel). This function can be used to request that Email Main
enable the button specified by the argument, ButtonName. If the button is enabled - whether
it was already disabled or is disabled as a result of the function call - the function will return
TRUE, otherwise it will return FALSE. The calling component should check on whether the
function call succeeded and proceed accordingly.
*/

em_ReturnValue emh_enableButton (
                str             ButtonName
)
```

API Functions Required Implementation of all Component Types

```
/*************************************************************
**/
/* emc_startMeUp
Used by Email Main to tell an installable component to start. This function will execute prior
to initialization of the component's data structures. Which should only be intialized after the
component receives the emc_initWindow call from Email Main.
This function is necessary for the Director/Lingo implementation.
*/

em_ReturnValue emc_startMeUp (
)


/*************************************************************
**/
/* emc_initWindow
Used by Email Main to tell an installable component to initialize it's data structures and
prepare its graphical display. The component is passed the username of the current user. If
it requires additional user information in order to initialize, it can call emh_getUserInfo
within it's implementation of this function.
*/

em_ReturnValue  emc_initWindow (
        em_UserName
)


/*************************************************************
**/
/* emc_closeWindow
Used by Email Main to tell an installable component to free all memory that it has used, close
it's window, and shut down.
*/

em_ReturnValue emc_closeWindow (
)


/*************************************************************
**/
/* emc_getComponentInfo
Used by Email Main to get required information such as componentName, componentID, etc.
from the installable component.
*/

em_ComponentInfo emc_getComponentInfo (
)
```

API Functions required of a Mailbox Handler Component

```
/*************************************************************
**/
/* mbx_getMessageNumbers
```

```
Used by Email Main to get the message number of the currently selected message in the
mailbox browser. If no message is selected, the list should be empty.
*/

list of int mbx_getMessageNumbers (
)


/*****************************************************************
**/
/* mbx_getMessage
Used by Email Main to get the message data structure of the message with
em_MessageNumber from the mailbox currently displayed in the mailbox browser. If the
function fails, e.g. if there is no message with the given message number, the function returns
an empty list.
*/

em_MailData mbx_getMessage (
        em_MessageNumber
)


/*****************************************************************
**/
/* mbx_trashMessages
Used by Email Main to tell the mailbox component to update it's display and it's data
structures to delete messages with messageNumbers in the argument list. If the function fails,
e.g. if one of the message numbers is invalid, the function returns FALSE, otherwise it returns
TRUE. This function should be implemented so that it does not perform partial deletes, i.e.
either it succeeds in deleting all of the messages in the list or it should not delete any message.
*/

em_ReturnValue mbx_trashMessages (
        list of em_MessageNumber
)
```

```
/***************************************************************
**/
/* mbx_openMailbox
Used by Email Main to tell the mailbox component to display the mailbox passed in the
argument.
*/

em_ReturnValue mbx_openMailbox (
        em_Mailbox
)
```

Functions required of a Message Handler Component

```
/**************************************************************
**/
/* msh_sendMessage
Used by Email Main to tell a message handling component to pass back a fully completed
message data structure so that it can be sent to the recipient specified in the message's address
field. The message handling component should update it's display as appropriate for a
message that has been sent. It should also change it's state to #display mode because a
message that has already been sent should not be editable. If the function fails, e.g. if a fully
completed message cannot be constructed (for example, if the user has not specified a
message recipient), the function returns an empty list.

The message handling component will normally control all dialogs with a user that pertain to
the message under construction. For example to alert the user to the fact that a message
recipient is required. However, if the message handling component fails to pass back a
properly formatted, completed message data structure, (or an empty list acknowledging
failure) Email Main will detect the error and alert the user about the field or fields that have
not been specified.
*/

em_MailData  msh_sendMessage (
)


/**************************************************************
**/
/* msh_openMessage
Used by Email Main to pass a message data structure to a message handling component so
that it can be displayed. The message handling component should display the message in the
specified mode - either #author or #display. If the em_Mode argument is #display the
message should not be editable. Otherwise the message should be opened so that it can be
edited.

If the function fails, e.g. if an error is detected in the message body, the message handler
returns FALSE, otherwise the message handler returns TRUE.
*/

em_ReturnValue  msh_openMessage (
        em_MailData
        em_Mode
)




/**************************************************************
**/
/* msh_replyMessage
Used by Email Main to inform a message handling component to display the currently active
message for editing as a reply. In order to reply the message handing component will
generally create a new message with the mode set to #author. The new message body may
contain material from the original message that is being replied to. In addition, message
handling components that handle different player roles may enable or disable various role
specific tools at this time. For example, the Rebus message handler will change the
RebusState of the new message and enable guessboxes as appropriate.
```

```
If the function fails, e.g. if an error is detected in the message body, the message handler
returns FALSE, otherwise the message handler returns TRUE.
*/

em_ReturnValue  msh_replyMessage (
)


/*****************************************************************
**/
/* msh_clearMessage
Used by Email Main to inform a message handling component that the current message
should be cleared from the display and from the message handling component's data
structures. This function is used, for example, when the user indicates they want to trash the
current message by clicking on the "trash" button in the Email Main purple panel.

If the function fails, the message handler returns FALSE. Otherwise the message handler
returns TRUE.
*/

em_ReturnValue  msh_clearMessage (
)
```

```
/*************************************************************
**/
/* msh_printMessage
Used by Email Main to inform a message handling component that a message should be
printed. This function is used, for example, when the user indicates they want to print the
current message by clicking on the "print" button in the Email Main purple panel.
When the argument, em_mailData, is an empty list, the message handler component should
print the currently active message. Otherwise the message handler component should print
the message argument. Normally, if the message handler component has been fully
initialized and is displayed in a window, Email Main will call this function with an empty list
for an argument.

The function may also be used by the Main Email program to have a message handler print a
message even though the message handler component has not been fully initialized and
displayed in a window. For example, this will occur if an active mailbox component receives
a print request from Email Main for a message that has been selected in the mailbox browser.
In this case, Email Main will send a request to the appropriate message handler component to
print the message without fully starting it up and initializing its window. Therefore the
message handler should implement the msh_printMessage function so that the following
sequence of function calls succeeds - emc_startMeUp, msh_printMessage(message).

If the function fails, the message handler returns FALSE. Otherwise the message handler
returns TRUE.
*/

em ReturnValue  msh_printMessage (
        em_MailData
)
```